Patent Application
Attorney Docket No.: 09469.010001; 97.0005

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **METHOD AND SYSTEM FOR INITIALIZING A KEY MANAGEMENT SYSTEM**

APPLICANTS: **Chui-Shan Teresa LAM**
Jameel ur Rahman SYED

"EXPRESS MAIL" Mailing Label Number: EV 014256346 US
Date of Deposit: December 21, 2001

METHOD AND SYSTEM FOR INITIALIZING A KEY MANAGEMENT SYSTEM

Background of Invention

[0001] Connection of company networks to the Internet has resulted in a requirement for increased network security. This has resulted in some software systems, *i.e.* Financial Software Systems, requiring security information as part of their operation. Security information includes such information as encryption/decryption keys for encrypted databases, administrator passwords to access external resources such as a directory server, etc. Security information is typically stored in a configuration file that is accessible by the software systems.

[0002] While security information is used to ensure the security of particular software systems and the data they use, security information itself is typically not secure. Security information typically stored in a configuration file is in "clear text" *i.e.* text in ASCII format. However, some software systems include devices to safeguard security information, such as, storing security information in encrypted form. While this will protect the security information when it is not being used, it does not protect the security information when it is accessed. When security information is accessed by a particular software system, the security information must first be decrypted prior to being used. During this time the security information is in clear text and thus is exposed.

[0003] To protect security information, software system operators typically use Hardware Security Modules (HSM). HSM physically protect security information by providing tamper-resistant security information storage. Additionally, HSM perform functions such as security information generation, security information

backup, security information management, etc. Security information inside an HSM is typically accessed by a proprietary software interface. Figure 1 illustrates a typical network system using HSM.

[0004] Typically, the network system has a security server (10) that contains root security information. The root security information allows a user with that information to access any resource on the network. Security information used by the security server is stored on an HSM (12). Additionally, the security server (10) runs the HSM interface that allows the security server to control all HMS (12) on the network. The network system as also typically has an application server (14) and a directory server (16). The application server (14) typically runs the business logic for a server-based application. The directory server typically runs a directory service. The directory service enables a user to locate hosts and services on a network. The application server (14) and the directory server (16) also both store security information in HSM (12). Finally, the network system contains a web server (18). The web server (18) typically runs server software that uses HyperText Transfer Protocol to serve up HyperText Mark-up Language documents and any associated files and scripts when requested by a client, such as a Web browser (20). The web server (18) stores security information in a HSM (11).

[0005] When a web browser (20) is running a web application via a web server (18) and an application server (14), there may be a request to access sensitive data such as a financial data located on the application server (14). The application server (14) on behalf of the web browser (20) will request some security information from the HSM (12). The security information on the HSM (12) is then accessed via the HSM Interface (11).

Summary of Invention

[0006] In general, in one aspect, the invention relates to a network system for key management, comprising: a server, a key management system providing process logic for key management system initialization located on the server, a key management system storage providing a secure data storage for the key management system, and an interface providing a means for inputting data into the key management system.

[0007] In general, in one aspect, the invention relates to a network system for key management, comprising: a server, a key management system providing process logic for key management system initialization located on the server, a key management system storage providing a secure data storage for the key management system, an interface providing a means for inputting data into the key management system, and a client computer operatively connected to the server, wherein the client computer comprises a user interface to input data into the key management system.

[0008] In general, in one aspect, the invention relates to a method for initializing a key management system comprising: entering data into a key management system interface, entering a key encryption key into the key management system interface, combining data into a tuple, encrypting the tuple with the key encryption key to produce a secret token, storing the secret token in a vector, hashing the key encryption key, storing a hashed key encryption key in the vector, storing a list of keys in the vector, serializing the vector to produce a serialized file, and storing the serialized file in a key management system storage.

[0009] In general, in one aspect, the invention relates to a method for initializing a key management system comprising: entering data into a key management system interface, entering a key encryption key into the key management system interface, combining data into a tuple, encrypting the tuple with the key encryption

key to produce a secret token, storing the secret token in a vector, hashing the key encryption key, storing a hashed key encryption key in the vector, storing a list of keys in the vector, serializing the vector to produce a serialized file, storing the serialized file in a key management system storage, encoding a key field of the tuple, randomizing the order of the list of keys, randomizing the order of the secret tokens in the vector, and generating data to encrypt.

[0010] In general, in one aspect, the invention relates to an apparatus for initializing a key management system comprising: means for entering data into a key management system interface, means for entering a key encryption key into the key management system interface, means for combining data into a tuple, means for encrypting the tuple with the key encryption key to produce a secret token, means for storing the secret token in a vector, means for hashing the key encryption key, means for storing a hashed key encryption key in the vector, means for storing a list of keys in the vector, means for serializing the vector to produce a serialized file, means for storing the serialized file in a key management system storage, means for encoding a key field of the tuple, means for randomizing the order of the list of keys, means for randomizing the order of the secret tokens in the vector, and means for generating data to encrypt.

[0011] Other aspects and advantages of the invention will be apparent from the following description and the appended claims.

Brief Description of Drawings

[0012] Figure 1 illustrates a typical network system using Hardware Security Modules.

[0013] Figure 2 illustrates a typical network system in accordance with one or more embodiments of the present invention.

[0014] Figure 3 illustrates a Key Management System in accordance with one or more embodiments of the present invention.

[0015] Figure 4 illustrates a 3-tuple in accordance with an embodiment of the present invention.

[0016] Figure 5 illustrates a vector in accordance with an embodiment of the present invention.

[0017] Figure 6 illustrates a typical graphical user interface to input data, in one or more embodiments of the present invention.

[0018] Figure 7 illustrates, in flow chart form, the typical steps involved in initializing a Key Management System.

[0019] Figure 8 illustrates a typical 4-tuple used in another embodiment of the present invention.

[0020] Figure 9 illustrates a vector in accordance with another embodiment of the present invention.

Detailed Description

[0021] In the following detailed description of the invention, numerous specific details are set forth in order to provide a more thorough understanding of the invention. However, it will be apparent to one of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known features have not been described in detail to avoid obscuring the invention.

[0022] The present invention relates to initializing a key management system. Further, the present invention relates to a method for securing keys within the key management system. Further, the present invention relates to a method for storing keys within the key management system.

[0023] Figure 2 illustrates an exemplary network system in accordance with one or more embodiments of the present invention. The network system typically includes a web server (22), and a client computer (28) containing a web browser (30). Additionally, the web server (22) is typically operatively connected to an application server (32) containing a Key Management System (KMS) (24) and a Key Management System Storage (KMS Storage) (26), and to a directory server (34). The web server (22) and the web browser (30) typically communicate using Hypertext Transfer Protocol (HTTP) running over Transport Control Protocol/ Internet Protocol (TCP/IP). The KMS (24) contains process logic, and the KMS storage (26) provides a secure data storage location *e.g.,* a hard drive, where information processed by the KMS (24) is stored. Further, the KMS storage (26) is secured by the operating system via file permissions.

[0024] In one or more embodiments of the present invention, the KMS storage (26) is located on a different computer than the KMS (24). Further, the computers are connected via a secure connection such as a connection using 128-bit encryption running over a Secure Socket Layer (SSL).

[0025] While the present invention is described in according to one embodiment those skilled in the art will appreciate that the KMS (24) and KMS storage (26) may be distributed across the network system.

[0026] Figure 3 illustrates a Key Management System (KMS) in accordance with one or more embodiments of the present invention. The KMS (24) includes a memory (36), a hashing module (38), an encryption module (42), an encoding module (40), and a serialization module (46). The memory cache (36) is a random access memory (RAM) subsystem in which frequently used data values are duplicated for quick access. Specifically, within the KMS (24) the memory(36) stores data initially sent to the KMS (24) prior to processing. The hashing module (38) hashes a Key Encryption Key (KEK) within the KMS (24). By applying a

hashing function to a piece of data, the resulting data is a reorganized version of the original data. Additionally, hash functions used to secure data are typically designed such that they can not be "reverse engineered." The purpose of hashing the KEK is to provide a means to store the KEK in a secure format. In one embodiment of the present invention, MD5 is used as the hash function. Those in the art will appreciate that different hashing functions may be employed, *e.g.*, MD2, MD4, etc.

[0027] The encryption module (42) contains encryption tools. The encryption tools include tools for key generation and tools for encryption. The key generation tools typically use a randomly generated seed as part of the input to the key generation tool. Specifically, within the KMS (24), the encryption module (42) includes process logic that generates the random seed for input to the key generation tool. The generated keys are used to access secure systems and/or applications such as a directory server. The encryption tool obtains data to be encrypted from the memory(36), or KMS generated keys from the key generation tools within the encryption module (42), and encrypts them using a specified encryption function. The encryption module (42) uses a KEK as the encryption key. In one embodiment of the present invention, the KEK includes three distinct parts: a KEK Pin, a KEK Salt, and a KEK Iteration. Constraints may be placed on all three parts of the KEK. For example, the KEK Pin may have a requirement to be at least 10 characters with a minimum of 3 numbers (N) and 3 letters (L) *e.g.*, KEK Pin: NNNLNLLNLN. Depending on the encryption algorithm used, the KEK may have greater or fewer than three parts. Additionally, some encryption algorithms may also apply constraints to individual KEK parts. For example, in one embodiment of the present invention uses PBE with MD5 in combination with Triple Data Encryption Standard (DES) as the encryption algorithm. This imposes a constraint on the KEK salt by limiting the KEK salt to eight characters. Further, the KEK Iteration is limited to an integer. Those skilled in the art will appreciate

that different encryption algorithms or combination of encryption algorithms may be used in conjunction with the present invention.

[0028] In one embodiment of the present invention, the encryption module uses a symmetric algorithm e.g., Triple Data Encryption Standard (DES). Symmetric algorithms rely on encryption and decrypting data using the same encryption key. Thus, a KEK is used to encrypt and decrypt data within the KMS.

[0029] In another embodiment of the present invention, the encryption module uses an asymmetric algorithm such as Rivest-Shamir-Adleman (RSA) algorithm. RSA creates a pair of related keys, one key to encrypt data and another key to decrypt data. Thus, a KEK may be used to encrypt the data and a corresponding Key Decryption Key (KDK) may be used to decrypt the data. The KEK and KDK are created simultaneously using the same algorithm *e.g.,* RSA algorithm. The key generation tools within the encryption module generate the KEK and KDK using random input either generated by the user or by the key generation tool.

[0030] The encoding module (40) converts the data into a binary representation that is 64 bit encoded. Typically, encoding is performed to hide the clear text names of the data being encrypted. The serialization module (46) obtains input from the hashing module (38), the encoding module (40)and encryption module (42) and stores it in a vector. Once all the data has been received, the serialization module (46) processes the vector to create a serialized file. Serialization is the process of saving an object's state to a sequence of bytes, such that it may be rebuilt into a live object at some future time. Specifically, within the KMS the vector is converted into an object within the serialization module (46) and subsequently converted into a serialized file. The serialized file allows the data within the vector to persist beyond the time the KMS (24) is active. Additionally, the serialized file may be copied and transferred to another system where it may be

stored as a backup. The process of creating the serialized filed is typically carried out using a Java™ Serialization Application Program Interface (API).

[0031] The KMS (24) uses three main data structures: a tuple, a vector, and a serialized file. Figure 4 illustrates a 3-tuple in accordance with an embodiment of the present invention. The 3-tuple (47) includes three data fields: a key field (48), a value field (50), and a type field (52). The key field (48) contains an identifying name of a value, *e.g.,* Admin_Password. The value field (50) contains a value identified by the name in the key field (48). The type field (52) may contain either "USER" or "GENERATED." "USER" corresponds to a value in the value field (50) that was entered by the user. "GENERATED" corresponds to a value in the value field (50) that was generated by the KMS, specifically the key generation tool in the encryption module. The three fields are combined to produce a 3-tuple (47). Data input into the KMS is first stored as a 3-tuple (47) within the memory prior to processing

[0032] Figure 5 illustrates a vector in accordance with one or more embodiments of the present invention. In data structures, a vector is a one-dimensional array, where an array is a set of items arranged in a single column or row. Additionally, the vector data structure can also dynamically grow based on the size of the items within the set. Specifically, a vector (54) with a KMS includes three distinction portions: a secret token portion (56), a KEK Hash portion (58), and an encoded key portion (60). The secret token portion (56) contains all the encrypted 3-tuples, each encrypted 3-tuple is herein referred to as a secret token. In one embodiment of the present invention the secret tokens are stored in a random order within the secret token portion (56). Following the secret token portion (56) is the KEK Hash portion (58). The KEK Hash portion (58) holds the result of applying the hash function to the KEK. The KEK Hash is output from the Hashing Module. The final portion is the Encoded Key Portion (60), the encoded key portion (60)

contains an encoded list of Keys (from the Key field in the 3-tuple). In one embodiment each key is 64-bit encoded and separated by a semi-colon. In another embodiment, the keys are combined into a list and the list is subsequently encoded. For example, if there were three keys then the encoded key portion (60) may have the following content: ENCODED (Key_A; Key_B ;Key_C). Those skilled in the art will appreciate that the encoded keys may be separated in different manner. Encoded keys are output from the encoding module, and then stored in the vector (54). In one embodiment of the present invention, the list of Keys may be placed in a random order prior to encoding.

[0033] A serialized file is a flat-file. A flat-file is a file that consists of a single record type in which there is not embedded structure information that governs relationships between the records. Specifically, with serialized flat-files they are "flattened" by the Sun® Mircosystem's Java™ Serialization Application Program Interface (API) such that they may be rebuilt at some future time. The serialization module takes a vector as input and produces a serialized file as output.

[0034] Figure 6 illustrates a typical graphical user interface (GUI) to input data, in one or more embodiments of the present invention. The GUI (60) may be part of a stand alone application or integrated into a web browser. The GUI (60) provides the user with a means to input data into the KMS. A field name section (64) contains the key's *e.g.,* card.ldap.admin.dn, which are stored in the key field of the 3-tuple. A value input section (66) contains two input text boxes: a value text box (68) and a field text box (70). The user inputs a value corresponding to a key in the field name section (64) into the corresponding value text box (68), the user then re-enters the value in the confirm text box (70). In one embodiment of the present invention the text typed into the value text box (68) and the confirm text box (70) is displayed as "clear text." In another embodiment of the present

invention the text typed into the value text box (68) and the confirm text box (70) is displayed as a series of asterisks. A generate randomly section (72) contains a series of checkboxes, one for each key. The user may check a box for a given key, which prompts the KMS to generate a value for that particular key. As mentioned above, the KMS generates the values using a key generation tool within the encryption module.

[0035] Once the user enters data into the above mentioned sections the user then proceeds to enter in a Key Encryption Key (KEK) section (74) on the GUI (62). The KEK section (74) on the GUI contains text-input fields for each portion of the KEK that is required. For example, referring to Figure 6, the KEK section (74) requires the user to input three pieces of information: a KEK PIN (76), KEK Salt (78), and a KEK Iteration (80). Similar to entering data into the value input section (66), the KEK section also requires the user to enter the data twice. In one embodiment of the present invention the text typed into the KEK section, (74) is displayed as clear text. In another embodiment of the present invention the text typed into the KEK section, (74) is displayed as a series of asterisks.

[0036] Once all the data has been entered user may check the "Write KEK to file for automatic KMS Initialization" checkbox (82). By checking the "Write KEK to file for automatic KMS Initialization" checkbox, the KEK section (74) information will be saved in a file that is accessed when the KMS is used. If this box is not checked, then every time the KMS is started the user will have to enter the KEK section (74) information. After the user has made a decision to check the "Write KEK to file for automatic KMS Initialization" checkbox (82), the user may click a "Create File" Button (84) to input the data into the KMS for it to be processed.

[0037] Referring again to Figure 2, consider the following scenario. A network operator adds a directory server (34) to the companies existing network

infrastructure *e.g.* a web server (22) and application server (32). The application server (32) is running an security application that is used to verify remote web users. The security application stores user name and password information in encrypted form on the directory server (34) running a Lightweight Directory Application Protocol (LDAP) compliant directory service. Thus, the security application requires an administrator level username and password to logon onto the LDAP-compliant directory service running on the directory server (34). Additionally, the security application requires the decryption keys to decrypt the data on the directory server (34). The present invention allows all the required access data *i.e.,* administrator username, administrator password, decryption key to be stored on the web server (22) in a secure format. The user initially enters all the required access data into a KMS (26) GUI. The GUI may be integrated with a web browser (30). The web browser is typically running a secure connection such as Secure Socket Layer (SSL). The user then proceeds to create a serialized file which contains all the required data in a secure format and then stores the serialized file in the KMS storage (26).

[0038] Figure 7 illustrates, in flow chart form, typical steps involved in initializing a Key Management System. The user enters data into the KMS GUI (Step 100). For example, referring to the above scenario, a user may enter the administrator username, administrator password, etc. The user then enters a KEK into the GUI (Step 102). Once the user clicks the "Create File" Button, the KMS combines the data into 3-tuples as mentioned above (Step 104). The key field of a first 3-tuple is then stored in the memory (Step 106). The KMS then encrypts the 3-tuple using the KEK as the encryption key to produce a secret token (Step 108). The secret token is then stored in a vector (Step 110). If there are any more 3-tuples to encrypt (Step 112), the above steps are repeated. If there are no more 3-tuples to encrypt (Step 112), the KMS hashes the KEK with a specified hashing algorithm. The result of hashing the KEK is then stored in the vector (Step 114). The KMS

then combines the keys into a list, 64 bit encodes them and stores them in the vector (Step 116). The vector is then processed by the serialization module to produce a serialized file (Step 118). The serialized file is then stored in the KMS storage (Step 120).

[0039] In one embodiment of the present invention, the KEK is stored on a smart card and input into the KMS via a GUI that has an interface with a smart card reader.

[0040] In one embodiment of the present invention, the 3-tuple is converted to a 4-tuple allowing the KMS to scale to use with multiple application simultaneously. For example, consider the above scenario with the security application. If another application such as an accounting application is added to the application server, then a another set of data will be required *e.g.,* administrator name, administrator password, etc to access the accounting application and the accounting application data. Figure 8 illustrates a typical 4-tuple used in a KMS serving multiple applications. The 4-tuple (86) contains all three fields that the 3-tuple contains *e.g.,* a key field (88), a value field (90) and a type field (92) plus an additional field: an application name field (94). The application name field (94) contains the name of the application.

[0041] In addition to modification made to the 3-tuple, the content in the vector is also modified. Figure 9 illustrates a typical vector used in a KMS serving multiple applications. A modified vector (96) contains the same portions as the previous embodiment of the vector *e.g..,* a secret token portion (98), a key encryption key portion (100), and an encoded key portion (102). All content in the secret portion (98) of the modified vector (96) is tagged with the application name obtained from the application name field in the 4-tuple. The application name tag may be 64-bit encoded. The secret tokens for all applications are then stored in the modified vector. The content in the encoded key portion (102) of the modified vector (96)

is modified such that each key is tagged with an application name. All the tagged keys are then combined into a list, 64-bit encoded, and stored in the vector. The vector is then serialized as described above. The KMS GUI is modified to allow input of the application name.

[0042] Advantages of the present may include one or more of the following. In some embodiments, the present invention provides a software solution to key management system. Further, the present invention may be integrated into existing network infrastructure without requiring additional hardware. In some embodiments, the present invention is scalable to manage keys for multiple applications. In some embodiments, the present invention allows sensitive data to be readily backed-up and recovered. In some embodiments of the present invention, the keys are never stored as clear text. Further, the present invention allows the KMS to be distributed over multiple servers within a network system. Further, the present invention allows the KEK to contain multiple portions *e.g.,* salt, count, integer, such that KEK may be distributed to multiple security officers. Those skilled in the art can appreciate that the present invention may include other advantages and features.

[0043] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of this disclosure, will appreciate that other embodiments can be devised which do not depart from the scope of the invention as disclosed herein. Accordingly, the scope of the invention should be limited only by the attached claims.